A Hybrid Multimodel Neural Network for Nonlinear Systems Identification

I. Baruch*-**, F. Thomas***, R. Garrido*, and E. Gortcheva*
*CINVESTAV-IPN, Mexico D.F., MEXICO,
E-mail: baruch@ ctrl.cinvestav.mx
**IIT-BAS, Sofia, BULGARIA,
E-mail: ips@bgearn.acad.bg
***IRI-UPC, Barcelona, SPAIN,
E-mail: thomas@iri.upc.es

Abstract

An improved universal parallel recurrent neural network canonical architecture, named Recurrent Trainable Neural Network (RTNN), suited for state-space systems identification, and an improved dynamic back-propagation method of its learning, are proposed. The proposed RTNN is studied with various representative examples and the results of its learning are compared with other results,, given in the literature. For a complex non-linear plants identification, a fuzzy-rule-based system and a fuzzy-neural multimodel, are used. The fuzzy-neural multimodel is applied for mechanical system with friction identification.

1. Introduction

The Neural Network (**NN**) modelling and application to system identification, prediction and control was discussed for many authors [1], [2], [3], [4], [5]. Mainly, two types of **NN** models are used: Feedforward (**FFNN**) and Recurrent (**RNN**). There exists some drawbacks of all described in the literature **NN** models. As it could be seen in [2], [3], [5], there exists a great variety of **NN** models and a universality is missing. All **NN** models are sequential in nature as implemented for systems identification. The **FFNN** model uses one or two tap-delays in the input, [1], and **RNN** models usually are based on the autoregressive model architecture , [2], [5], which is one-layer sequential one. Some of the applied **RNN** models are not trainable - others are not trainable in the feedback part, [4]. Most of them are dedicated to a **SISO** and not to a **MIMO** applications, [2]. In more of the cases, the stability of the **RNN** is not considered, especially during the learning, [4]. In the case of **FFNN** application for systems identification, the plant model is described by means of four possible nonlinear models, [1]. The linear part of the plant model, especially the system order, has to be known and the **FFNN** approximates only the nonlinear part of it. The described in the literature learning methods for dynamic **NN**s are too complicated and difficult to understand, [3]. A new Jordan canonical **RNN** has been proposed by the authors in its previous paper, [6], and it has been extended to a multimodel case, [7]. They propose a Fuzzy-Neural (**FN**) approach, extending the Takagi-Sugeno model to resolve more complex identification tasks, [7]. The aim of this paper is to perform a deep topological analysis of the described models and to propose an improved hybrid **RTNN** model, introducing some weight feedback restrictions to preserve **RTNN** stability and to confirm its identification abilities. Simultaneously with the improvement of **RTNN** topology, some advanced researches has been done on the methods of its learning, leading to the design of a new **RTNN** dynamic Backpropagation (**BP**) learning method and to its new fuzzy-neural multimodel implementation. The proposed **RTNN** model for system identification is studied by means of various non-linear discrete-time dynamic objects. Three types of object models are suggested: models, non-linear on their output, state and input. Simulation examples of nonlinear objects and of mechanical object with friction, [8], identified by two **RTNN** models, coordinated by a rule-based fuzzy system (**RBFS**), are given.

2. Description of the RTNN Model and its Learning

The described in [6] two-layer Jordan canonical **RTNN** architecture, improved with an additional stability preserving restriction on the feedback weight matrix of the hidden layer, is given for both continuous and discrete time cases in the form:

0-7803-5529-6/99/$10.00 ©1999 IEEE

$$\dot{x} = Jx + Bu,\ z = S(x);\ y = S(Cz) \quad (1)$$
$$Re(J_i) < 0 \quad (2)$$

$$X(k+1)=JX(k)+BU(k),\ Z(k)=S[X(k)];\ Y(k)=S[CZ(k)] \quad (3)$$
$$|J_i| < 1 \quad (4)$$

where the two layer equations of **RTNN** are separated by semicolon; **y, x, u** (**Y, X, U**) are output, state and input vectors with dimensions **l, n, m,** respectively; **J = block-diag(J_i)** is a (**nxn**)- block-diagonal weight matrix; J_i are blocks of **J** with (**1x1**) or (**2x2**) dimensions (equations (2) and (4) are stability conditions, imposed on all blocks J_i of **J**, which are in fact conditions on system eigenvalues for both continuous and discrete-time cases); **B** and **C** are (**nxm**) and (**lxn**)- weight matrices; **S** (**x**) is a vector-valued activation function, given as:

$$S'(inp) = [s(inp_1), s(inp_2),.,s(inp_j), s(inp_n)] \quad (5)$$

Here **inp** is the input variable of the activation function $s(inp_j)$, which is an element of the vector function **S.** The activation functions in use are the sigmoid and the saturation function, given by the equations:

$$s(inp) = 1/[1+\exp(-inp)];\ inp=\sum_i (w_i x_i + w_{io}) \quad (6)$$

$$sat(inp) = \begin{cases} +1, & inp \geq +1 \\ inp, & 0 \leq inp < +1 \\ 0, & inp < 0 \end{cases} \quad (7)$$

where w_i, w_{io} are trainable weights of the **RTNN**; **S'** signifies a vector transpose of **S**. The saturation function is used as approximation of the sigmoid function to improve the **RTNN** architecture, facilitating its realisation.

The three layer versions of the **RTNN** continuous and discrete-time architecture are as follows:

$$\dot{x} = Jx + q,\ z = S(x);\ q = S(Bu);\ y = S(Cz) \quad (8)$$

$$X(k+1) = JX(k)+Q,\ Z(k)=S[X(k)];\ Q = S[BU(k)];\ Y(k)=S[Z(k)] \quad (9)$$

where **q** ,(**Q**) is a **n**- input vector of the hidden layer.

The given above **RTNN** model (in two and three layered versions) could be linearised and its dynamic behaviour could be studied by the first stability law of Liapunov, where the stability conditions, (2), (4), for continuous and discrete time state-space models, could be used. Than it is easy to analyse the **NN** model controllability, observability and identifiability. From the block structure of **B** and **C**, corresponding to the block structure of **J**, we can conclude if the **RTNN** could be learned or not, [6]. The main advantages of the proposed three layer Jordan Canonical Form (**JCF**) **RTNN** architecture, defined for both continuous and discrete-time cases are that the described **JCF RTNN** model is an universal hybrid neural model which contains one or two **FF** layers and one recurrent hidden layer with completely decomposed dynamics, as the matrix **J** is block-diagonal. So, it has a minimum number of parameters and it is completely parallel, as the Jordan canonical form is parallel with respect to the regressive model, which is a sequential one. The **RTNN** architecture is described in state-space form (**SISO** or **MIMO**) and could serve as one-step ahead state predictor/estimator. The **RTNN** model is non-linear in large and linear in small, so the matrices **J, B, C,** obtained as a result of learning, could be used for analytical design of linear state/output feedback/feedforward control laws. Finally, the **RTNN** could solve the optimal control problem itself by means of a **NN** mapping , [1]. The obtained **RTNN** model is a robust model, as the learning method applied for weights adjustment is a dynamic **BP** method, which is based on the network sensitivity model, [1]. It also permits to perform node pruning and weight fixing during the learning.

The most common used **BP** updating rule, applied for the two-layer **RTNN** canonical model, [6], is the following:

$$W_{ij}(k+1)=W_{ij}(k) + \eta\, \Delta W_{ij}(k) \quad (10)$$

where: W_{ij} is a general weight, denoting the **ij**-th weight element of each weight matrix (**C, J, B**) in the **RTNN** model to be updated; ΔW_{ij} , (ΔC_{ij}, ΔJ_{ij}, ΔB_{ij}), is the weight correction of W_{ij}; η is the learning rate parameter. If there are some oscillations in the error during the learning, a momentum term can be added in the weight updating rule (10). One of the following two forms can be used:

$$W_{ij}(k+1)=W_{ij}(k) + \eta\, \Delta W_{ij}(k) + \alpha\, \Delta W_{ij}(k-1) \quad (11)$$
$$W_{ij}(k+1)=W_{ij}(k) + \eta(1-\alpha)\, \Delta W_{ij}(k) + \alpha\, \Delta W_{ij}(k-1) \quad (12)$$

where α is a momentum term learning rate parameter. The equation (12) establish some inverse dependence between the learning parameters of the first and second updating terms. A lot of experiments of learning with different rates of learning η and α in equation (11), has been done. These experiments show that the optimal combination of these learning parameters is obtained when the following inequality condition yields:

$$r_{max} < sqrt(\eta^2+\alpha^2) < 1;\ r_{max} = \max|\lambda_i| \quad (13)$$

where $\max|\lambda_i|$ is the maximum eigenvalue of the identified plant model. The experimentally obtained rule (13) shows that the circle with radius **r**, which depends on the values

of η and α ($r=sqrt(\eta^2+\alpha^2)$), must contain all eigenvalues of the matrix **J** of the discrete-time **RTNN** model (2). The weight corrections of the updated matrices in the discrete-time **RTNN** model, described by eqn. (2), are performed using the following update equations:

- For the output layer:

$$\Delta C_{ij}(k) = [T_j(k)-Y_j(k)]\, Y_j(k)\, [1-Y_j(k)]\, Z_i(k) \quad (14)$$

where: ΔC_{ij} is the weight correction of the **ij**-th elements of the (**lxn**) learned matrix **C**; T_j is a **j**-th element of the target vector; Y_j is a **j**-th element of the output vector; Z_i is an **i**-th element of the output vector of the hidden layer.

- For the hidden layer:

$$\Delta B_{ij}(k) = R\, U_i(k) \quad (15)$$
$$\Delta J_{ij}(k) = R\, X_i(k-1) \quad (16)$$
$$R = C_i(k)\, [T(k)-Y(k)]\, Z_j(k)\, [1-Z_j(k)] \quad (17)$$

where: ΔB_{ij} is the weight correction of the **ij**-th elements of the (**mxn**) learned matrix **B**; C_i is a row vector of dimension (**1xl**), taken from the transposed matrix **C'**; **[T-Y]** is a (**lx1**) output error vector, through which the error is back-propagated to the hidden layer; U_i is an **i**-th element of the input vector **U**; X_i is an **i**-th element of the vector **X**; ΔJ_{ij} is the weight correction of the **ij**-th elements of the (**nxn**) block-diagonal matrix **J** under learning; **R** is an auxiliary variable. The same equation for **RTNN** learning may be applied for the continuous-time case, given by equation (1). Another improvement of the **RTNN** learning algorithm, successfully applied for the **BP** learning of discrete-time **RTNNs** consider unimportant units pruning and non-useful connections removing, which lead to exclusion of weights or nodes from the process of learning. The improved learning algorithm for **RTNN** was tested with several linear and non-linear dynamic objects. The topology improvements are also carefully studied. The applicability of the improved **RTNN** model for system identification and prediction is illustrated by appropriate example of non-linear dynamic system.

4. A Fuzzy System Rule Based Coordination Model

Let us assume that the unknown system $y = f(x)$ generates the data **y(k)** and **x(k)** measured at **k, k-1,..,** then the aim is to use this data to construct a deterministic function $y = F(x)$ that can serve as a reasonable approximation of $y = f(x)$ in which the function **f(x)** is unknown.

The variables $x = [x1,...,xp]' \in \aleph \subset \Re^p$ and $y \in Y \subset \Re$ are called regressor and regressand, respectively. In Fuzzy System (**FS**) modelling, the function **F(x)** is represented as a collection of if-than fuzzy rules of the type:

IF antecedent proposition **then** consequent proposition

The Takagi-Sugeno model, cited in [7], is a mixture between a linguistic and mathematical regression models. The rule antecedents describes the fuzzy regions in the input space. The rule consequent is a crisp mathematical function of the inputs. This mathematical function could be a dynamic function too, given by a state-space mathematical model of the form:

R_i: **If** x(k) **is** J_i and u(k) **is** B_i

$$\textbf{then} \begin{cases} x_i(k+1) = J_i\, x(k) + B_i\, u(k) \\ yi(k) \quad = C_i\, x(k) \end{cases} \quad (18)$$

Our proposal is that this function be a **RTNN** model, given by equation (2). So, the **FS** model obtains the form:

R_i: **If** x is J_i and u is B_i **then** $y_i = N_i(x)$, i=1,2,.., (19)

where the consequent function $y_i = N_i(x)$ represents the **RTNN** model given by the equation (2). To incorporate the **RTNN** model (2) into the **FS** model (19), it have to use the **RTNN** model (2) in the consequent proposition part of (19) and the restrictions, expressed by the saturation of (7) to construct the antecedent proposition part of (19). The biases, obtained in the process of **BP** learning of the **RTNN** model could be used to form membership functions, as they are natural centres of gravity, [7], for each variable. The number of rules could be optimised using the mean-square error (**MSE**) of **RTNNs** learning, which is a natural measure of precision of the approximation of the non-linear object sub-model (**MSE** is up to 3%). The structure of the entire identification system contains a fuzzyfier a Fuzzy Rule-Based System (**FRBS**), and a set of **RTNN** models. The system does not need a defuzzyfier, because **RTNN** models are crisp limited linear state-space models. A possible adaptive control system, could contain also a set of controllers incorporated in a **FRBS**, designed on the base of the obtained set of **RTNNs.**

5. Simulation Results

5.1. Example 1

The first example, taken from [3], is a discrete-time model of nonlinear object with input and output smooth nonlinearities, identified by one **RTNN** model. The learning signal, is chosen as sequence of pulses with random amplitude and width. The neural model quality is

verified comparing reactions of the plant and the **RTNN** to an unknown generalisation signal of one-epoch length. This signal, taken from [3], has the form:

$$u(k) = \sin(\pi k/25),\ 0 < k < 251;\ \ 1.0\ \ 250\ < k < 501;$$
$$-1.0\ \ 500\ < k < 751;\ \ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32)$$
$$+ 0.6\sin(\pi k/10),\ k < 1001 \qquad (20)$$

The **SISO** discrete-time nonlinear plant model is described by the following equations, [3]:

$$x_1(k+1) = x_2(k)$$
$$x_2(k+1) = -0.15\,x_1(k) + 0.8x_2(k) + h(k)$$
$$h(k) = 3\,u_1^3(k) / [1 + 4\,u_1^2(k)]$$
$$u_1(k) = 1.9\,u(k)$$
$$y(k+1) = \{[0.3\,x_1(k+1) + x_2(k+1)] / 3.5\}^3 \qquad (21)$$

Simulation results, obtained for **RTNN** identification model, using sigmoid activation function and **BP** learning algorithm with momentum term, are given in the **Fig.1**

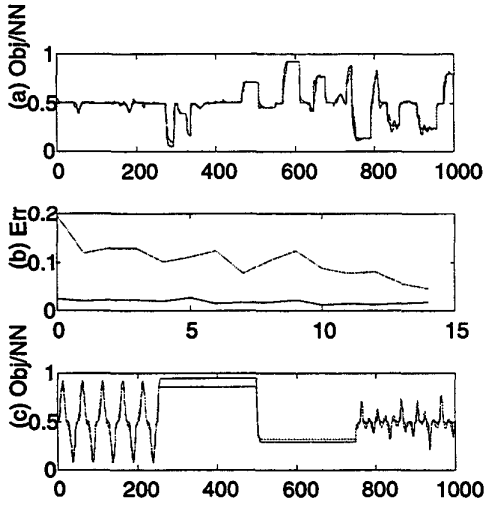

Fig.1. RTNN architecture **(1,7,1)**, η = **0.5**, α = **0.75**, sigmoid activation function, **MSEG = 4%, MSEL = 2%, N = 14 epochs.**

(a) Output of RTNN (dashed line) and Object (solid line) during last epoch of training (1000 iterations).

(b) RTNN MSEG error of generalisation (dashed line) and training (solid line) for all epochs of learning

(c) Output of RTNN (dashed line) and Object (solid line) during last epoch of generalisation.

The **RTNN** architecture **(1, 7, 1)** applied, has one input, seven neurones in the hidden layer and one output. The time of learning is **N = 14** epochs (one epoch contain **1000** iterations) and the final Mean Square Errors (**MSE**) of learning and generalisation obtained are **MSEL = 2%, MSEG = 4%,** respectively. The obtained results in [3] for the same examples are: time of learning between **62 000** and **77 000** iterations which is about **5** times longer with respect to **RTNN**. The means square error, obtained in [3] for architecture **(3:0)** is **3.45%** and for architecture **(6:1)** is **6.41%**. Note that in [3], an **FFNN** architecture with memory neurones is used, which duplicates the number of neurones required. So the first architecture (the better one) uses **1** network neurone and **1** memory neurone in the input layer, **3** network neurones plus **3** memory neurones in the hidden layer, and **1** network neurone and **0** memory neurones in the output layer. The total number of neurones required for this NN is **9**, and for **RTNN** it is **8**.

5.2. Example 2

The second example is a discrete-time model of nonlinear **1- DOF** mechanical object with friction, taken form [8]. The discrete-time model of the **1-DOF** mass mechanical system with friction, is given in the form:

$$x_1(k+1) = x_2(k)$$
$$x_2(k+1) = -0.025\,x_1(k) - 0.3\,x_2(k) + 0.8\,u(k) - 0.1\,fr(v,k) \qquad (22)$$

where: $x_1(k)$, $x_2(k)$ are system states; **k** is a discrete time variable and the friction force **fr(v,k)**, [8], is assumed to be modelled as follows:

$$fr(v,k) = F_{slip}(v)\,a(v) + F_{stick}(u)[1 - a(v)] \qquad (23)$$

$$a(v) = \begin{cases} +1, & |v(k)| > \alpha \\ \quad ; \alpha > 0 & \\ 0, & |v(k)| \le \alpha \end{cases} \qquad (24)$$

The sticking friction provides the value of the friction forces at zero velocity **v**. The term is used to describe whether the mass will stick or break free from the static friction forces. The positive and negative limits on the static friction forces are given by F_s^+ and F_s^-, respectively. Generally, they are not equal in magnitude, and the model should consider these asymmetry. It is modelled as follows:

$$F_{stick}(u) = \begin{cases} F_s^+, & u(k) \ge F_s^+ \\ u(k), & F_s^- < u(k) < F_s^+ \\ 0, & u(k) \le F_s^- \end{cases} \qquad (25)$$

The mass cannot move until the applied force is greater in magnitude than the respective static friction force. The slipping function $F_{slip}(v)$ provides values of the friction at non-zero velocity and is represented by:

$$F_{slip}(v) = F_d^+(v)\,b(v) + F_d^-(v)\,b(-v) \qquad (26)$$

$$b(v) = \begin{cases} 1, & v(k) > 0 \\ 0, & v(k) \le 0 \end{cases} \quad (27)$$

$$F_d^+(v) = F_s^+ - \Delta F^+ [1 - \exp(-v/v_{cr}^+)] + \beta^+ v \quad (28)$$

$$F_d^-(v) = F_s^- - \Delta F^- [1 - \exp(-v/v_{cr}^-)] + \beta^- v \quad (29)$$

where ΔF^+ and ΔF^- are the respective drops from the static to the kinetic force level; v_{cr}^+ and v_{cr}^- are the critical Stribeck velocities, and β^+ and β^- are the viscous friction coefficients. The friction force is modelled as a summation of the Coulomb friction, viscous friction, and the Stribeck effect. The Stribeck effect is that the friction force is decreasing with increasing fluid lubrication. Some models consider dynamic lag effect of the friction force with respect to the velocity, which effect could be neglected. For sake of simplicity, some slip friction parameters for both velocity directions could be considered as equal (e.g. $v_{cr}^+ = v_{cr}^- = v_{cr}$; $\beta^+ = \beta^- = \beta$). The **1-DOF** mechanical system, [8], is consider to have the following friction parameters: α **= 0.001 m/s;** F_s^+ **= 4.2 N ;** F_s^- **= - 4.0 N;** ΔF^+ **= 1.8 N ;** ΔF^- **= - 1.7 N ;** v_{cr} **= 0.1 m/s;** β **= 0.5 Ns/m.** Simulation results, obtained for this plant, identified by two **RTNN** models, coordinated by a **FRBS**, are given in **Fig.2** and **Fig.3**. The epoch time is **100** iterations and the learning signal is a sum of sinusoids with different frequencies, as follows:

$$u(k) = \sin(\pi k/25),\ 0 < k < 26;$$
$$1.0\ \ 25 < k < 51; -1.0\ \ 50 < k < 76;$$
$$0.3 \sin(\pi k/25) + 0.1 \sin(\pi k/32) +$$
$$0.6 \sin(\pi k/10),\ 75 < k < 101 \quad (30)$$

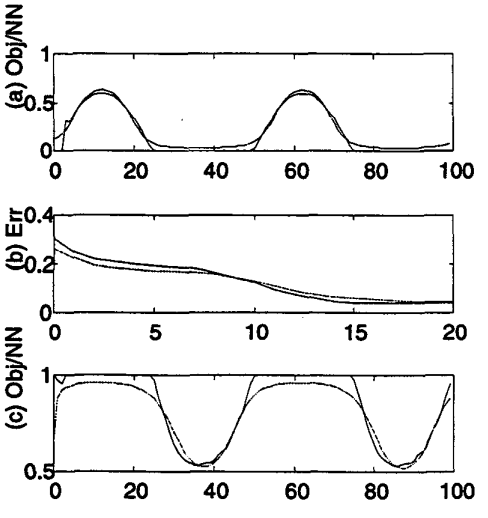

Fig.2. Two **RTNN** model approach, η **= 0.2,** α **= 0.9,** sigmoid activation function

*(a) Output of the first (positive) **RTNN** (dashed line) and **Object** (solid line) during last epoch of learning (100 iterations).*

*(b) **RTNN MSE** of learning the second **RTNN** (dashed line) and learning the first **RTNN** (solid line) for all epochs. Time of learning - 20 epochs. MSE for last epoch of learning - 3%.*

*(c) Output of the second (negative) **RTNN** (dashed line) and **Object** (solid line) during last epoch of learning (100e iterations.).*

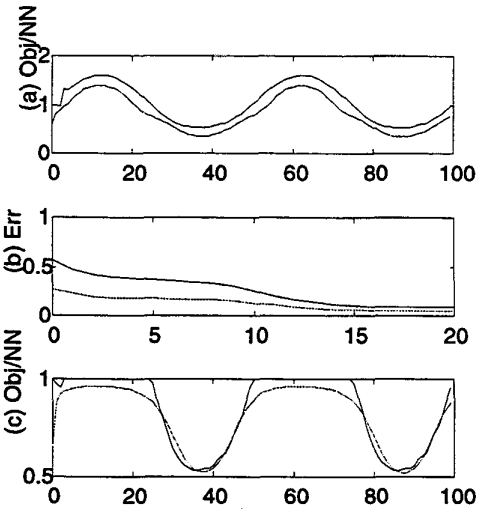

Fig.3. Two **RTNN** model approach, η **= 0.2,** α **= 0.9,** sigmoid activation function

*(a) Output of the combined fuzzy-neural model (dashed line) and **Object** (solid line) during last epoch of learning (100 iterations).*

*(b) **RTNN MS** error of learning the second **RTNN** (dashed line) and learning the combined fuzzy-neural model (solid line) for all epochs. Time of learning - 20 epochs. MSE for last epoch of learning - 3%.*

*(c) Output of the second (negative) **RTNN** (dashed line) and **Object** (solid line) during last epoch of training (100 iterations.).*

The first and the third graphics of **Fig.2.** show the training results of both **RTNN** models during the last epoch of training (for positive and negative output signals, separately). It compare the positive / negative output of the system with the output of the corresponding **RTNN** model. The second graphics shows the Mean-Square-Error (**MSE**) of training for both **RTNN** models during all the period of training, given in epochs. The results show an excellent 20 epochs convergence with **3% MSE** for both models. The first graphics of **Fig.3.** show the training results of the combined fuzzy-neural multi-model during the last epoch

of training. It compare the output of the system with the output of the combined neural multi-model. For sake of comparison, the third graphics shows again the results given in **Fig.2.c.** for the second (negative output) **RTNN**. The second graphics of **Fig.3.** shows the **MSE** of the combined and the second models during all the period of learning. The results show an excellent **20 epochs** convergence with the same **3% MSE.**

6. Conclusions

An improved universal parallel recurrent neural network canonical architecture, named Recurrent Trainable Neural Network , suited for state-space systems identification, and an improved dynamic back-propagation method of its learning, are derived. The obtained hybrid **RTNN** model is studied with a representative example and the results of its learning and generalisation are compared with other results, given in the literature. The comparison shows the better performance of the **RTNN**. For a complex non-linear plants identification, a fuzzy-rule-based system and a fuzzy-neural multimodel, are used. The fuzzy-neural multimodel is applied for **1-DOF** mass mechanical system with friction identification. The result obtained for two **RTNN** model identification of this complex object also shows a good performance of both **RTNN** during the learning.

Acknowledgements

This work has been partly supported by the CINVESTAV-IPN, Mexico D.F., MEXICO, IRI-UPC, Barcelona, SPAIN and NSF, MEST, grant 611/1996, Sofia, BULGARIA.

References

1. K.S.Narendra, and K. Parthasarathy, Identification and Control of Dynamic Systems using Neural Networks, *IEEE Transactions on NNs, 1(1), 1990, 4-27.*
2. J.T.Connor, R. Martin, and L. Atlas, Recurrent NN and Robust Time Series Prediction. *IEEE Transactions on NNs, 5(2), 1994, 240-253.*
3. P.S.Sastry, G. Santharam, and K.P. Unnikrishnan, Memory Networks for Identification and Control of Dynamical Systems. *IEEE Transactions on NNs, 5(2), 1994, 306-320.*
4. D.T.Pham,. and S. Yildirim, Robot Control using Jordan Neural Networks, *Proc. of the Internat. Conf. on Recent Advances in Mechatronics, Istanbul, Turkey, Aug. 14-16, 1995,* (O. Kaynak, M. Ozkan, N. Bekiroglu, I. Tunay, Eds., Bogaziçi University Printhouse, Istanbul, Turkey, 1995), *Vol. II, 888-893.*
5. A.C.Tsoi, and A.D. Back, Locally Recurrent Globally Feedforward Networks: A Critical Review of Architectures. *IEEE Transactions on NNs, 5(2), 1994, 229-239.*
6. I.Baruch, I. Stoyanov, and E. Gortcheva, Neural Network Models of Dynamic Processes: Stability and Learning, *Proc. of the 3-rd Int. Symp. MMAR'96, Sept. 10-13, 1996, Miedzyzdroje, Poland,* (S. Banka, S. Domek, Z. Emirsajlov, Eds., TU of Szczecin, Poland, 1996), *Vol. 3, pp. 1169-1174.*
7. I.Baruch, and E.Gortcheva, Fuzzy-Neural Model for Nonlinear Systems Identification, *Proc. of the 5-th IFAC Workshop AARTC'98, April 15-17, 1998, Cancun, Mexico,* (D.G.Nocetti, J.S. Gonzalez, P.A. Contla, P.J.Fleming, Eds., AMCA, Mexican Society of Automatic Control,1998), *283-288.*
8. Seon - Woo Lee, and Jong-Hwan Kim, Robust Adaptive Stick-Slip Friction Compensation, *IEEE Transactions on IE, 42(5), 1995, 474-479.*